一、准备

1. 注册一个WinDHP账号

请在WinDHP数字健康平台的WinDHP用户中心注册页 上注册账户；

拥有账户后即可在WinDHP用户中心登录页 进行登录。

注册账号之后会得到WinDHP平台分配的唯一的一对**AppKey**、**AppSecret**；

2. 订购产品

要对接联调WinDHP的某个产品的API，您需要先在WinDHP云市场 中搜索、订购这个产品;

2.1 搜索录入关键词

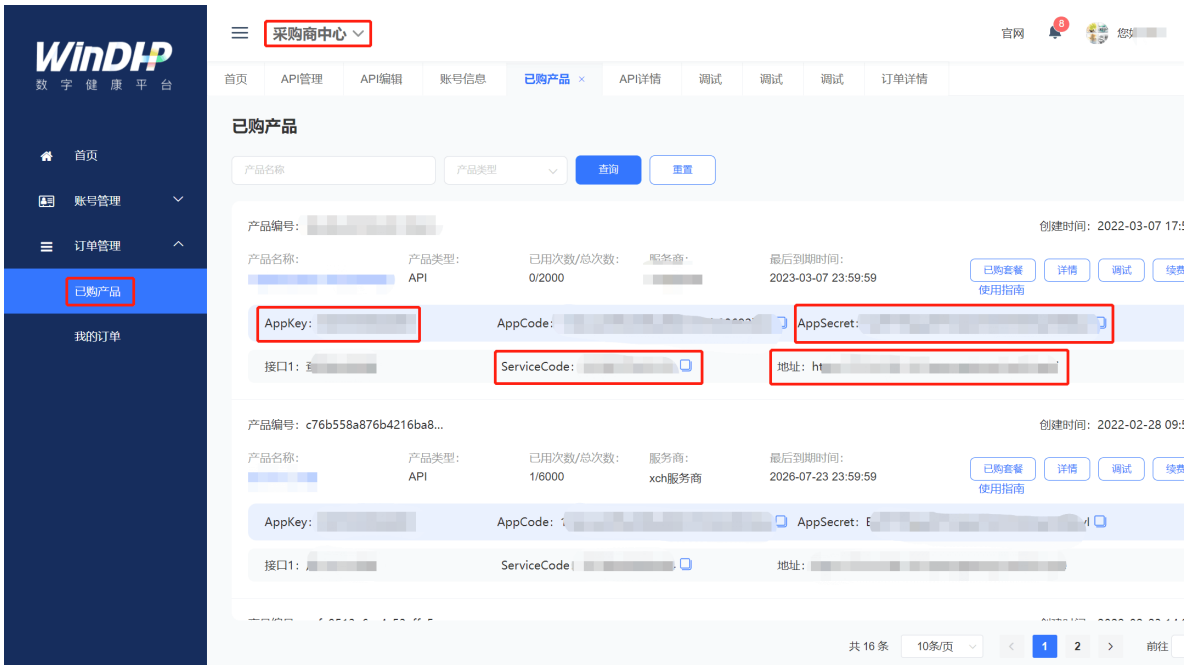
关键词如：医院名称、产品名称关键词等

2.2 搜索结果

2.3 订购/订阅产品

点击搜索到的产品，跳转到产品详情页，核实产品符合自己的需求后，进行订购/订阅操作即可；

订购订阅之后，即可在WinDHP用户中心-采购商中心的 订单管理-已购产品 菜单页看到已购产品：$\color{#FF0000}{注意：如果是订购的是医院相关服务，需通知对应机构进行授权后，才能看到已购产品}$



3. 签名认证三要素 AppKey、AppSecret、ServiceCode

在已购产品圈中的元素：

> **AppKey** *注册账户后分配给WinDHP用户的，和AppSecret结合使用*
>
> **AppSecret** *注册账户后分配给WinDHP用户的，签名认证秘钥和AppKey结合使用*
>
> **ServiceCode** *每个接口的唯一码，采购商调用接口时使用*
>
> **地址** *固定不变 https://openapi.windhp.com/call/simple*

AppKey、**AppSecret**、**ServiceCode**在签名认证中都会使用到，我们简称之为三要素；

二、概述

> WinDHP采购商在注册、登录、订购之后，便可以调用已订购产品下的每个接口；
>
> WinDHP平台的产品接口在被调用时，需要遵守WinDHP平台签名认证规则，此规则具有防越权、防重放、防篡改等防护功能；
>
> WinDHP签名认证规则要求采购商请求已订购产品接口，基于HTTP请求；

对接WinDHP产品接口，HTTP请求头必须包含：

请求头key	请求头value
X-Service-Code	请求头X-Service-Code，取值三要素之一 **ServiceCode**
X-Ca-Key	请求头X-Ca-Key，取值三要素之一 **AppKey**
X-Ca-Nonce	请求头X-Ca-Nonce，请求唯一标识，每次请求保证唯一，通常使用UUID即可
X-Ca-Timestamp	请求头X-Ca-Timestamp，发起请求时的时间戳，值为当前时间的毫秒数（即从1970年1月1日到当前的间隔时间的毫秒数）
X-Conent-MD5	请求头X-Conent-MD5，对请求参数先进行MD5摘要再进行Base64编码获取摘要字符串；具体参考(三-1)章节：请求头X-Conent-MD5详解
X-Ca-Signature	按照特定规则对特定的HTTP请求头，进行签名算法计算的结果； 具体参考(三-2)章节：请求头X-Ca-Signature详解

三、请求头X-Conent-MD5、X-Ca-Signature详解

1、请求头X-Conent-MD5 详解

> 对于HTTP请求，分为POST、GET、DELETE、PUT请求类型；

1.1 参数预处理

1.1.1 POST请求body参数预处理

针对POST请求的body参数，一些特殊字符（空格、制表符\t、回车\r、换行\n）去除处理;

> 注意：如果 POST请求的body参数为空，则默认为"";

java代码示例：

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * 特殊字符（空格、制表符\t、回车\r、换行\n）的正则Pattern
 */
private static final Pattern pattern = Pattern.compile("\\s*|\t|\r|\n");

/**
 * POST请求的body参数替换方法
 * @param postBody post请求的body参数
 * @return 替换后的body参数
 */
public static String replaceSpecialCharacters(String postBody) {
    //匹配特殊字符
    Matcher m = pattern.matcher(postBody);
    //特殊字符替换为空字符
    return m.replaceAll("");
}
```

1.1.2 GET/DELETE请求参数预处理

1. 针对于GET/DELETE请求的QueryParam参数,会把**QueryParam参数对**,对参数key按照字典顺序重排序;
2. 然后拼接为key1=value1&key2=value2&key3=value3这种格式的字符串

> 注意：如果GET/DELETE请求的QueryParam参数为空，则默认为"";

java代码示例

```
/**
 * 读取queryParams,转换成key1=value1&key2=value2的格式
 * @param queryParams query参数对
 * @return key1=value1&key2=value2格式的参数字符串
 */
public static String resolveQueryParamsFromRequest(Map<String, String>
queryParams) {
    String[] keys = queryParams.keySet().toArray(new String[0]);
    Arrays.sort(keys);
    List<String> queryList = Lists.newArrayListWithExpectedSize(keys.length);
    for (String name : keys) {
        queryList.add(String.format("%s=%s", name, queryParams.get(name)));
    }
    return StringUtils.join(queryList, "&");
}
```

1.2 对参数MD5摘要以及Base64编码得到X-Conent-MD5的值

Http请求的参数经过预处理后，便可以进行MD5摘要和Base64编码，得到的结果就是**请求头X-Conent-MD5的值**

java代码示例：

```
import java.io.UnsupportedEncodingException;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

/**
 * 先进行MD5摘要再进行Base64编码获取摘要字符串；
 * @param bytes 字符串
 * @return 摘要，即请求头X-Conent-MD5的值
 */
public static String base64AndMD5(String param) {
    if (param == null) {
        return null;
    }
    try {
        byte[] bytes = param.getBytes(StandardCharsets.UTF_8.toString());
        final MessageDigest md = MessageDigest.getInstance("MD5");
        md.reset();
        md.update(bytes);
        return Base64.getEncoder().encodeToString(md.digest());
    } catch (final UnsupportedEncodingException e) {
        throw new RuntimeException(e.getMessage(), e);
```

```
    } catch (final NoSuchAlgorithmException e) {
        throw new IllegalArgumentException("unknown algorithm MD5");
    }
}
```

2、请求头X-Ca-Signature 详解

2.1 签名串规则

参与签名的签名串规则如下：(三者之间有换行符\n)

```
HTTPMethod
Content-Type
headerStrToSign（带“X-”前缀的请求头组装而成）
```

> **HTTPMethod** Http请求的类型，取值：POST/GET/DELETE/PUT

> **Content-Type** Http请求头Content-Type的值，例如：application/json; charset=utf-8

> **headerStrToSign** 组装规则:
> 带“X-”前缀的请求头参数Key转成小写字母， 按照**字典排序**后使用如下方式拼接：

- header 名转换为小写，跟上“:”英文冒号字符。
- 然后附加 header 值。
- 如果不是最后一条需构造签名的 header，以”&”符号进行连接。

headerStrToSign例子: x-ca-key:62989828116480&x-ca-nonce:68c694e0852542a88483635cd0b7cd04&x-ca-timestamp:1646710852847&x-content-md5:wZ5JeGLyko75WOxB0Ix06g==&x-service-code:41563211440128

最后把三者按照签名串规则组装： (注意换行符\n)

```
POST\napplication/json; charset=utf-8\nx-ca-key:62989828116480&x-ca-nonce:68c694e0852542a88483635cd0b7cd04&x-ca-timestamp:1646710852847&x-content-md5:wZ5JeGLyko75WOxB0Ix06g==&x-service-code:41563211440128
```

2.2 HmacSHA256签名算法得到X-Ca-Signature的值

按照签名串规则拼装好后，对加签串使用HmacSHA256签名算法加签，签名计算需要**三要素之一AppSecret**

java代码示例：

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public static void main(String[] args) {
    String stringToSign = "POST\n" +
        "application/json; charset=utf-8\n" +
```

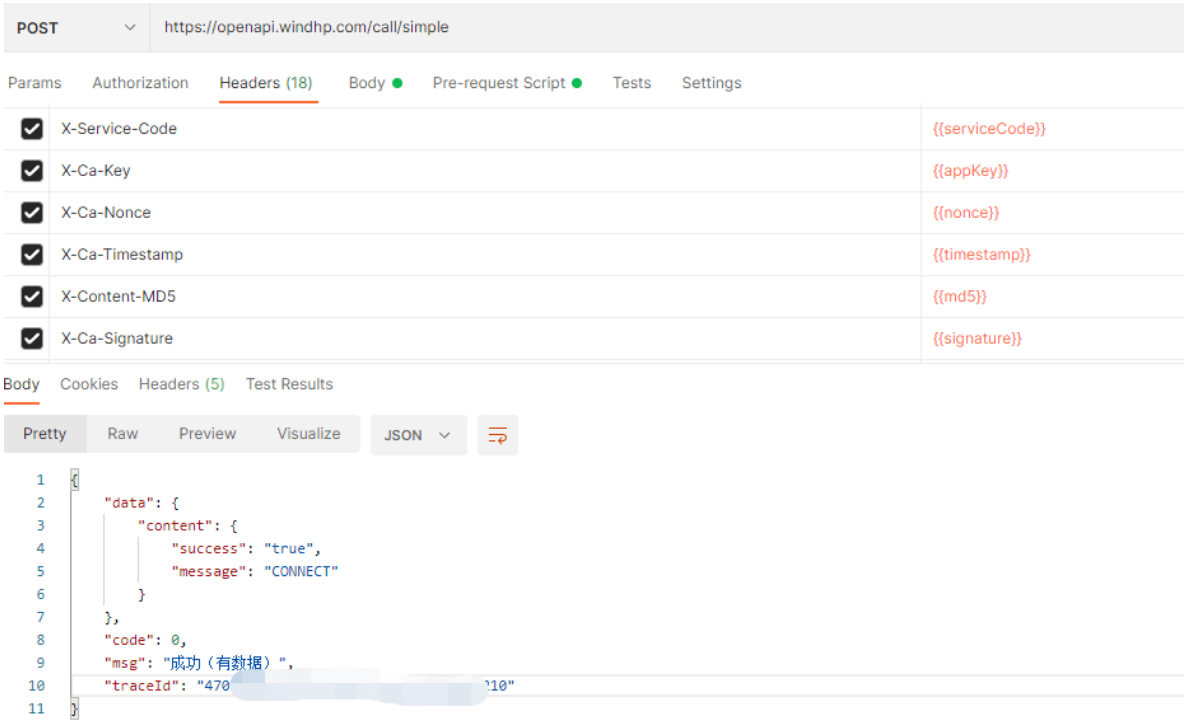
```
        "x-ca-key:62989828116480&x-ca-nonce:68c694e0852542a88483635cd0b7cd04&x-
ca-timestamp:1646710852847&" +
        "x-content-md5:wZ5JeGLyko75WOxBOIxO6g==&x-service-code:41563211440128";
    //appSecret即三要素之一AppSecret
    String appSecret = "XXXXXX";

    //xCaSignature 即签名得到的请求头X-Ca-Signature的值
    String xCaSignature = hmacSHA256(stringToSign, appSecret);
    System.out.println(xCaSignature);
}

private static final String ALGORITHM_NAME = "HmacSHA256";

/**
 * HmacSHA256签名算法
 * @param stringToSign 签名内容
 * @param appSecret 即三要素之一AppSecret
 * @return 签名结果
 */
public String hmacSHA256(String stringToSign, String appSecret) {
    try {
        Mac hmacSha256 = Mac.getInstance(ALGORITHM_NAME);
        SecretKeySpec secret_key = new SecretKeySpec(accessKeySecret.getBytes(),
ALGORITHM_NAME);
        hmacSha256.init(secret_key);
        return
DatatypeConverter.printBase64Binary(hmacSha256.doFinal(stringToSign.getBytes(Sta
ndardCharsets.UTF_8)));
    } catch (NoSuchAlgorithmException e) {
        throw new IllegalArgumentException(e.toString());
    } catch (InvalidKeyException e) {
        throw new IllegalArgumentException(e.toString());
    }
}
```

签名结果即请求头参数X-Ca-Signature的值；

- 签名错误排查方法
  当签名校验失败时，API网关会将服务端的签名内容放到HTTP应答的Header中返回到客户端，Key为：`X-Ca-Error-Message`，只需要将本地计算的签名内容与服务端响应头返回的`X-Ca-Error-Message`进行对比即可找到问题；

四、postman请求示例

1、postman请求示例



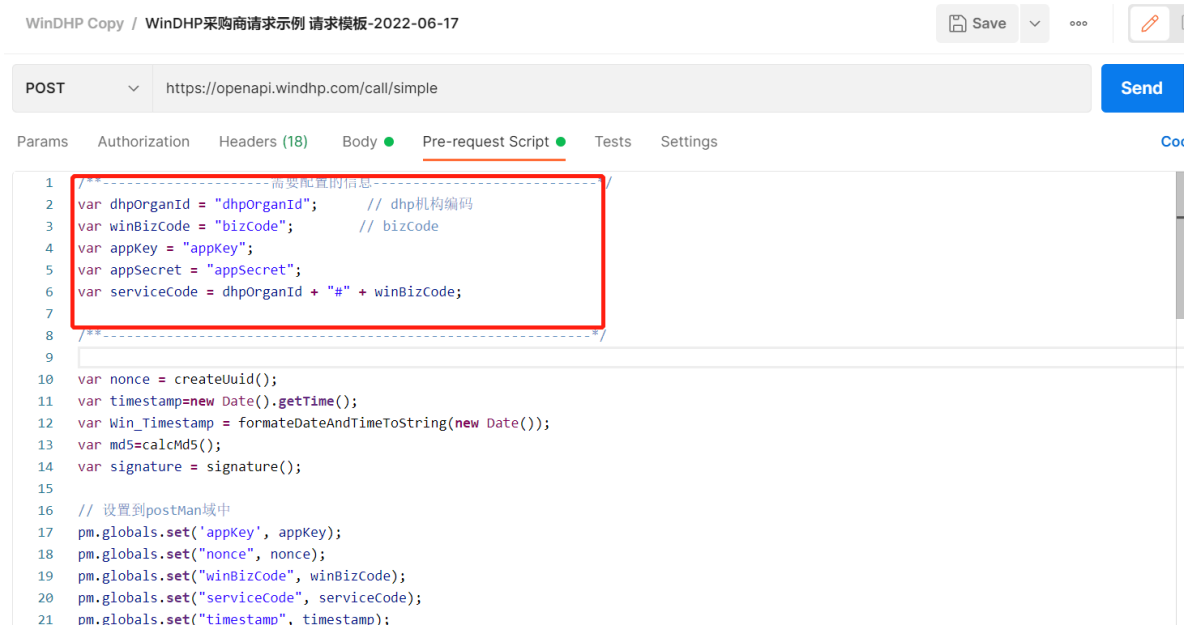
2、使用postman自动化脚本联调测试

> 为了方便采购商对接WinDHP平台，我们准备了postman自动化脚本：**WinDHP采购商请求示例脚本**；

> 可以根据录入的三要素AppKey、AppSrcret、ServiceCode,自动计算出请求头X-Conent-MD5、X-Ca-Signature的值，

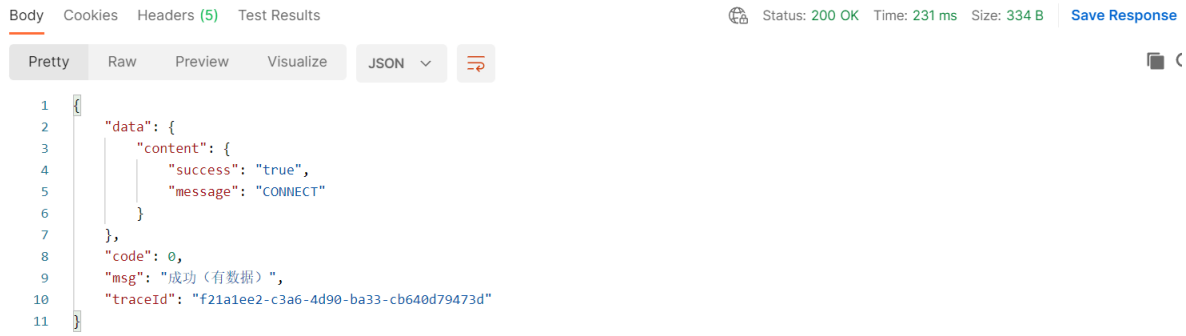
postman自动化脚本下载地址：
WinDHP采购商请求示例.postman_collection.json

脚本获取之后，直接导入到postman，只需要把“需要配置的信息” 配置成自己的即可开始使用：

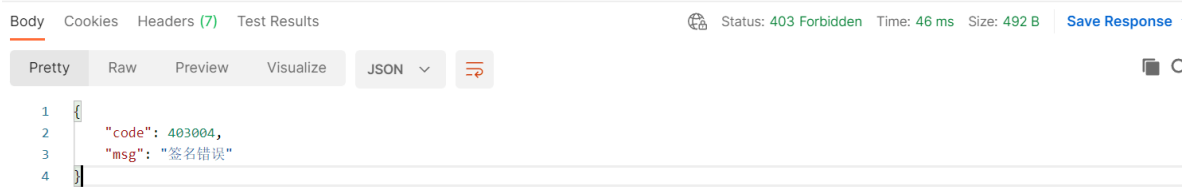


3、返回示例

3.1 成功返回示例：



3.2 错误返回示例：



3.3 错误代码

错误代码	HTTP状态码	描述	解决方案
401000	401	认证失败：${Reason}	查看相关认证要素，修改后重试
403000	403	签名错误	签名不匹配，请参考API网关签名文档
403001	403	无权调用该服务，请检查是否有该服务权限	鉴权不通过
403002	403	ip地址非法，请检查是否在白名单内	请参考网关安全设置
403600	403	X-Ca-Key is required	检查X-Ca-Key，修改后重试
403602	403	X-Ca-Timestamp is required	检查X-Ca-Timestamp，修改后重试
403603	403	X-Ca-Nonce is required	检查X-Ca-Nonce，修改后重试

错误代码	HTTP状态码	描述	解决方案
403604	403	X-Ca-Signature is required	检查X-Ca-Signature，修改后重试
403605	403	X-Content-MD5 is required	检查X-Content-MD5，修改后重试
403606	403	X-Service-Code is required	检查X-Service-Code，修改后重试
403610	403	Invalid X-Ca-Key, please check	检查X-Ca-Key，修改后重试
403611	403	Invalid X-Service-Code, please check	检查X-Service-Code，修改后重试
403612	403	contentMd5 error,please check	MD5 值计算错误，修改后重试
403613	403	时间超时，请确保时间戳在有效时间内	`X-Ca-Timestamp` 头中提供的时间戳已过期
403614	403	nonce can't be repeated	检测到请求重放，请求的 `X-Ca-Nonce` 头重复
403620	403	购买的次数已用完	购买的次数已用完
403621	403	购买次数已过期	购买次数已过期
403622	403	用户已欠费，请尽快充值续费	请尽快充值续费
404000	404	${Resource} 未找到	资源未找到
408000	408	访问服务超时	建议重试
429000	429	服务请求过多，已触发限流，请稍后再试	建议稍后重试
429001	429	请求次数过多，请稍后再试	建议稍后重试
429002	429	服务已触发系统保护，请稍后再试	建议稍后重试
500000	500	内部错误	建议重试
500999	500	服务内部错误	RPC调用错误
500601	500	服务商配置错误，请联系服务商	联系服务商或工单联系工作人员
503000	503	服务不可用/无法找到${ApplicationName}服务	建议稍后重试
503600	503	服务不可用，请联系服务商	服务商服务不可用，建议稍后重试
503601	503	服务降级了，请稍后再试	建议稍后重试
504000	504	服务请求超时	建议稍后重试

五、使用WinDHP SDK for Java工具包对接

> WinDHP SDK for Java按照WinDHP的签名规则，进行了封装，使用者只需要（AppKey, AppSecret, ServiceCode, 地址）；
> 即可完成对产品下接口的请求调用；
> 这里向您介绍如何获取 WinDHP SDK for Java 并开始使用。

1、安装依赖

作为采购商，使用 WinDHP SDK for Java，只需要添加依赖`windhp-openapi-java-sdk-client`。

获取 windhp-openapi-java-sdk-client-1.0.2.jar:
windhp-openapi-java-sdk-client-1.0.2.jar下载链接

通过Maven来管理项目依赖(推荐)

如果您使用Apache Maven来管理Java项目，只需在项目的`pom.xml`文件加入相应的依赖项即可。

但首先需要把windhp-openapi-java-sdk-client-1.0.2.jar包，使用如下命令安装到自己私有Maven仓库；

```
mvn install:install-file -DgroupId=com.winning -DartifactId=windhp-openapi-java-sdk-client -Dversion=1.0.2 -Dpackaging=jar -Dfile=D://windhp-openapi-java-sdk-client-1.0.2.jar
```

> -Dfile(jar包所在路径);

然后您只需在`pom.xml`中声明以下依赖即可：

```
<dependency>
    <groupId>com.winning</groupId>
    <artifactId>windhp-openapi-java-sdk-client</artifactId>
    <version>1.0.2</version>
</dependency>
```

还需要将以下这些依赖项添加到`pom.xml`文件中，否则将报告 NoClassDefFoundError 异常

```
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.15</version>
</dependency>

<dependency>
    <groupId>org.bouncycastle</groupId>
    <artifactId>bcprov-jdk15on</artifactId>
    <version>1.69</version>
</dependency>

<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.9.2</version>
</dependency>
```

2、快速使用

以下这个代码示例向您展示了调用 WinDHP SDK for Java 的3个主要步骤：

1. 创建IProfile实例并初始化。
2. 创建API请求并设置参数。
3. 发起请求并处理应答或异常。

> 注意，调试使用的okhttp3的version:4.9.2,低版本可能会报错。

```
package com.winning;

import com.winning.constant.Constants;
import com.winning.constant.SystemHeader;
import com.winning.exceptions.ClientException;
import com.winning.exceptions.ServerException;
import com.winning.profile.DHPProfile;
import com.winning.profile.IProfile;
import com.winning.request.Response;

public class Main {
    public static void main(String[] args) {
        IProfile profile = DHPProfile.getProfile(
                "<your-service-code>",    // 购买的产品的ServiceCode
                "<your-app-key>",         // WinDHP采购商账号的AppKey
                "<your-app-secret>");     // WinDHP采购商账号的AppSecret
        try {
            Response response = DHPHttpClient.get(profile)
                    .url("https://openapi.windhp.com/call/simple")
                    .addHeader(SystemHeader.CONTENT_TYPE,
Constants.APPLICATION_JSON)
                    .build()
                    .execute();
            System.out.println(response.string());
        } catch (ServerException e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        } catch (ClientException e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

通过响应头Response-header的`X-Trace-Id`或`X-Ca-Error-Message`，帮助开发者快速定位问题；